



# Desenvolvimento de Serviços e APIs

Faculdade Senac Pelotas

Escola de Tecnologia da Informação

Prof. Edécio Fernando Iepsen

Trabalhando em fluxos de trabalho críticos acionados por alterações em seu banco de dados?

## ORM

### VISÃO GERAL

- Introdução
- Prisma ORM em sua pilha
- Bancos de dados

### ESQUEMA PRISMA

- Visão geral
- Modelo de dados
- Introspecção
- Extensões PostgreSQL

Preview

### CLIENTE PRISMA

- Instalação e configuração
- ▼ **Consultas**
  - CRUD
  - Selecione os campos
  - Consultas de relação

Home / ORM / Cliente Prisma / Consultas

# Transações e consultas em lote

Uma transação de banco de dados refere-se a uma sequência de operações de leitura/gravação com *garantia de sucesso ou falha como um todo*. Esta seção descreve as maneiras pelas quais a API do cliente Prisma oferece suporte a transações.

## #Visão geral das transações #

### ! INFORMAÇÕES

Antes do Prisma ORM versão 4.4.0, não era possível definir níveis de isolamento nas transações. O nível de isolamento na configuração do seu banco de dados sempre é aplicado.

Os desenvolvedores aproveitam as garantias de segurança fornecidas pelo banco de dados agrupando as operações em uma transação. Essas garantias são frequentemente resumidas usando a sigla ACID:

- **Atômico** : garante que *todas ou nenhuma* operação das transações seja bem-sucedida. A transação foi *confirmada* com sucesso ou *abortada e revertida* .
- **Consistente** : Garante que os estados do banco de dados antes e depois da transação sejam *válidos* (ou seja, quaisquer invariantes existentes sobre os dados sejam mantidas).

- **Isolado** : garante que as transações executadas simultaneamente tenham o mesmo efeito como se estivessem sendo executadas em série.
- **Durabilidade** : Garante que, após o sucesso da transação, todas as gravações sejam armazenadas de forma persistente.

Embora haja muita ambigüidade e nuances em cada uma dessas propriedades (por exemplo, a consistência pode realmente ser considerada uma *responsabilidade no nível do aplicativo* em vez de uma propriedade do banco de dados ou o isolamento é normalmente garantido em termos de níveis de isolamento mais fortes e mais fracos ), no geral elas servem como uma boa diretriz de alto nível para as expectativas que os desenvolvedores têm ao pensar em transações de banco de dados.

"As transações são uma camada de abstração que permite a um aplicativo fingir que certos problemas de simultaneidade e certos tipos de falhas de hardware e software não existem. Uma grande classe de erros é reduzida a uma simples interrupção de transação, e o aplicativo só precisa tentar de novo." [Projetando aplicativos com uso intensivo de dados ↗](#) , [Martin Kleppmann ↗](#)

## Sobre transações no Prisma

O Prisma Client oferece as seguintes opções para usar transações:

- [Gravações aninhadas](#) : use a API do cliente Prisma para processar múltiplas operações em um ou mais registros relacionados dentro da mesma transação.
- [Transações em lote/em massa](#) : processe uma ou mais operações em massa com `updateMany`, `deleteMany`, e `createMany`.
- A `$transaction` API no cliente Prisma:
  - [Operações sequenciais](#) : passam um array de consultas do Prisma Client para serem executadas sequencialmente dentro de uma transação, usando `$transaction<R>(queries: PrismaPromise<R>[]): Promise<R[]>`.
  - [Transações interativas](#) : passe uma função que pode conter código do usuário, incluindo consultas do cliente Prisma, código não Prisma e outro fluxo de controle a ser executado em uma transação, usando `$transaction<R>(fn: (prisma: PrismaClient) => R, options?: object): R`

## A [#API](#) [\\$transaction](#) #

A `$transaction` API pode ser usada de duas maneiras:

- [Operações sequenciais](#) : passe um array de consultas do Prisma Client para serem executadas sequencialmente dentro de uma transação.

```
$transaction<R>(queries: PrismaPromise<R>[]): Promise<R[]>
```

- [Transações interativas](#) : passe uma função que pode conter código de usuário, incluindo consultas do cliente Prisma, código não Prisma e outro fluxo de controle a ser executado em uma transação.

```
$transaction<R>(fn: (prisma: PrismaClient) => R): R
```

### [#Operações sequenciais do cliente Prisma](#) #

A consulta a seguir retorna todas as postagens que correspondem ao filtro fornecido, bem como uma contagem de todas as postagens:

```
const [posts, totalPosts] = await prisma.$transaction([
  prisma.post.findMany({ where: { title: { contains: 'prisma' } } }),
  prisma.post.count(),
])
```

# Controle de Vendas de Farmácia

Você foi contratado para criar uma API para controlar as vendas dos medicamentos de uma farmácia.

A API deve manipular as seguintes models (tabelas):

- Medimento(s): contendo atributos como: id, nome, laboratorio, preco, controlado (sim/não), quantidade e quantMinima (que é o ponto de reposição).
- Cliente(s): contendo atributos como: id, nome, email, cidade, dataNasc ...
- Venda(s): contendo atributos como: id, clienteld, data, total (inicialmente 0).
- ItemVenda(s): contendo os atributos como: id, vendald, produtold, quantidade e preco

\* A transação deve envolver a inclusão de um item da venda, que deve atualizar a quantidade em estoque e aumentar o total da nota (venda).



# Controle de Vendas de Farmácia

- Defina as models e rode a *migration* e o *generate*.
- Insira (via MySQL) 3 ou 4 registros nas tabelas de clientes e medicamentos.
- Crie os arquivos (e rotas) para realizar o CRUD das vendas e itens da venda.
- Teste o funcionamento das transações
- Crie uma rota para enviar um e-mail para o cliente, que receba como parâmetro na URL o id da venda e construa uma tabela HTML com os itens vendidos ao cliente (deste id/nota) enviando-a no corpo do e-mail. Ou seja, nossa API vai permitir enviar para o e-mail do cliente a nota da venda.

